비트패턴을 기반으로 한 고속의 적응적 가변 블록 움직임 예측 알고리즘

신동식[†] · 안재형[††]

요 약

본 논문에서는 비트패턴을 기반으로 한 고속의 적응적 가변 블록 움직임 예측 알고리즘을 제안한다. 제안된 방법은 블록 내의 평균값을 기준으로 8bit 화소값을 0과 1의 비트패턴으로 변환한 후 블록의 움직임 예측을 수행한다. 비트변환을 통한 영상의 단순화는 움직임 추정의 계산적 부담을 감소시켜 빠른 탐색을 가능하게 한다. 그리고 블록 내의 움직임 정도를 미리 판별하여 이를 기반으로 한 적응적 탐색이 불필요한 탐색을 제거하고 움직임이 큰 블록에서는 정합 과정을 심화시켜 보다 빠르고 정확한 움직임 예측을 수행한다. 본 제안된 방식을 가지고 실험한 결과, 한 프레임 당 적은 수의 블록으로 고정된 크기의 블록을 가진 전역 탐색 블록 정합 알고리즘(full search block matching algorithm; FS-BMA)보다 예측 에러를 적게 발생시켜 평균 0.5dB 정도의 PSNR 개선을 가져왔다.

Fast Variable-size Block Matching Algorithm for Motion Estimation Based on Bit-pattern

Dong-Shik Shin[†] and Jae-Hyeong Ahn[††]

ABSTRACT

In this paper, we propose a fast variable-size block matching algorithm for motion estimation based on bit-pattern. Motion estimation in the proposed algorithm is performed after the representation of image sequence is transformed 8bit pixel values into 1bit ones depending on the mean value of search block, which brings a short searching time by reducing the computational complexity. Moreover, adaptive searching methods according to the motion information of the block make the procedure of motion estimation efficient by eliminating an unnecessary searching of low motion block and deepening a searching procedure in high motion block. Experimental results show that the proposed algorithm provides better performance-0.5dB PSNR improvement-than full search block matching algorithm with a fixed block size.

1. 서 론

최근의 과학 기술의 진보와 고속의 네트워크 환경은 폭넓은 멀티미디어 데이터 통신을 가능하게 하고 있다. 그러나 큰 대역과 큰 저장매체를 필요로 하는 멀티미디어 데이터의 특성과 실시간 처리를 요구하는 사용자들의 욕구는 아직까지도 지금의 통신환경 속에서 충분히 만족되지 못하고 있는 현실이다. 이에 따른 멀티미디어 데이터의 표준화 작업과 압축 기술에 대한 연구가 활발히 진행되고 있다. 그 중에 하나가 동영상 압축이다. 이는 낮은 비용으로 고화질의 동영상을 제공하는 것을 목적으로 한다. 동영상 압축은 동영상의 높은 시간적 상관관계의 특성을 이용한다. 따라서 프레임간의 시간적 중복성을 제거하는 움직임 예측은 동영상 압축의 가장 핵심적인 부분이다. 이 움직임 예측에는 간단하고 효과적인 방식인 블록 정합 알고리즘(block matching algorithm; BMA)[1]이 주로 사용된다. 이 방식은 이미 H.261/263,

† 준회원, 충북대학교 정보통신공학과 석사과정
†† 정회원, 충북대학교 정보통신공학과 교수

MPEG1/2에 채택되어져 있는 알고리즘이다. 블록 정합 알고리즘은 한 블록 내의 모든 화소들이 동일한 움직임을 갖는다는 가정 하에 한 블록 당 하나의 움직임 벡터(motion vector; MV)를 할당한다. 블록의 크기가 커지면 커질수록 더 적은 수의 움직임 벡터 전송으로 높은 압축률을 얻을 수 있지만 많은 예측 에러의 발생으로 화질은 떨어지게 된다.

블록 정합이라 함은 이전 프레임의 탐색 영역 안에서 현재 프레임의 현재 블록과 가장 유사한 블록을 찾는 과정이다. 전역 탐색 블록 정합 알고리즘(full search block matching algorithm; FS-BMA)은 탐색영역 내에서 탐색블록을 화소 단위로 이동시키면서 전부 검사하는 방식으로 최적의 성능을 나타내는 블록 정합 알고리즘이다. 그러나 이 알고리즘의 많은 계산적 부담은 예측의 정확도는 다소 떨어지더라도 탐색 속도의 개선을 가져다 줄 수 있는 three-step 탐색[2], Cross 탐색 알고리즘[3]과 같은 알고리즘들의 동기가 되어왔다. 반면 블록의 크기가 고정일 때 영상의 국부적인 변화에 따른 움직임을 효과적으로 반영하기 어렵기 때문에 이를 개선하기 위한 화질 개선 측면의 가변 블록 크기(variable block size)모델도 연구되어 왔다[4]. 이는 전역 탐색 블록 정합 알고리즘 보다 더 적은 예측 에러값을 발생시키지만 부가적인 계산적 부담이 따른다.

본 논문에서 제안하는 블록 정합 알고리즘은 블록 내의 평균값을 기준으로 8bit 화소값을 0과 1의 1bit로 변환한 후 움직임 예측을 수행하도록 한다. 비트 변환을 통한 영상의 단순화는 움직임 추정의 계산적 부담을 감소시켜 빠른 탐색을 가능하게 한다. 그리고 블록 내의 움직임 정도를 미리 판별하여 이에 따른 여러 적응적 탐색이 불필요한 탐색을 제거한다. 반면 움직임이 큰 블록에서는 정합과정을 심화시켜 보다 빠르고 정확한 움직임 예측을 수행한다.

본 논문의 구성은 다음과 같다. 2장, 3장에서는 제안 알고리즘에 대해 설명하고, 4장에서는 성능을 평가하며, 5장에서 결론을 맺는다.

2. 블록 정합 알고리즘

블록 정합 알고리즘은 현재 프레임을 $N \times N$의 겹쳐지지 않는 블록으로 분할한 후, 분할된 각각의 블록에 대해 이전 프레임에서의 그 블록과 가장 유사한 블록을 찾아내어 움직임 벡터를 얻어내는 방법이다. 그림 1은 두 프레임 사이의 블록 정합과정을 나타낸다. 두 프레임 사이에서 블록이 움직일 수 있는 최대 변위가 w일 때, 이전 프레임에서 탐색되어질 블록들은 $(N+2w)\times(N+2w)$ 크기의 탐색영역 안에 존재한다. 블록 정합이 수행될 탐색위치의 수는 $(2w+1)^2$이다.

(a) Reference Frame (b) Current Frame

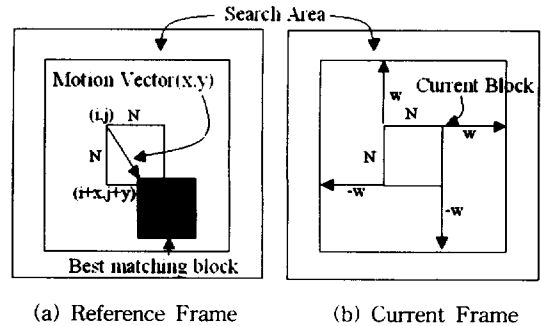



그림 1. 전역 탐색 블록 정합 방법

정합 블록 결정은 각각의 탐색 지점에서 식 (1)과 같은 두 블록 사이의 차분인 DBD(Displaced Block Difference)를 계산하여 이루어진다.

$$DBD(x,y) = \frac{1}{N\times N}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|F_n(i,j)-F_{n-1}(i+x,j+y)|$$
$$x,\ y \in [-w,\ -w+1,\ \ldots,\ w-1,\ w] \quad (1)$$

탐색영역 내의 후보 블록들 중 DBD를 최소로 하는 블록을 정합된 블록으로 선택하고, 그 블록의 수직/수평방향의 변위 (x, y)를 움직임 벡터(MV)로 결정한다. 여기서 N은 블록의 너비와 높이이다. F_n은 현재 프레임에서 예측되어질 블록을 나타내고 F_{n-1}은 이전 프레임에서 탐색영역 내에 있는 후보 블록이다.

2.1 비트패턴을 기반으로 한 블록 정합 알고리즘

비트패턴은 해당 블록과 탐색영역의 8bit 화소값을 1bit로 표현한 이진 영상을 의미한다. 이는 탐색과정에서의 계산량을 줄여 빠른 탐색이 이루어지도록 하기 위한 것이다. 탐색블록과 탐색영역의 비트패턴 B는 식 (2)를 통해서 얻어진다[5].

$$B(i,j) = \begin{cases} 1, & \text{if } p(i,j)\geq M \\ 0, & otherwise \end{cases} \quad (2)$$

여기서 $p(i,j)$는 탐색블록 또는 탐색영역의 (i,j)번째 화소값을 의미하고, M은 탐색블록의 평균 화소값을 의미한다. 비트패턴은 각각의 블록에 대해서 직관적인 블록의 정보를 나타낸다. 즉, 탐색블록의 비트패턴이 탐색영역 내의 어떤 블록의 비트패턴과 매우 유사하다면 원영상에서의 두 블록도 유사한 이미지의 특징을 갖고 있다고 볼 수 있다. 그러나 많은 예측에러(prediction error)의 발생으로 이를 줄일 수 있는 방법이 뒤따라야 한다. 비트패턴을 기반으로 전역탐색을 하였을 경우 DBD 계산에서 FS-BMA경우에 비해 83%정도의 계산량을 줄일 수 있다. 이는 영상의 표현을 8bit에서 1bit로 단순화시켰기 때문이다. bit 수에 따른 한번의 DBD 계산을 위한 연산의 수를 구하는 수식은 식 (3)과 같다[6].

$$\begin{aligned} N(n) &= 32n + \sum_{k=0}^{7} 2^k (n+7-k)/8 \\ &= 32n + (255n + 247)/8 \end{aligned} \tag{3}$$

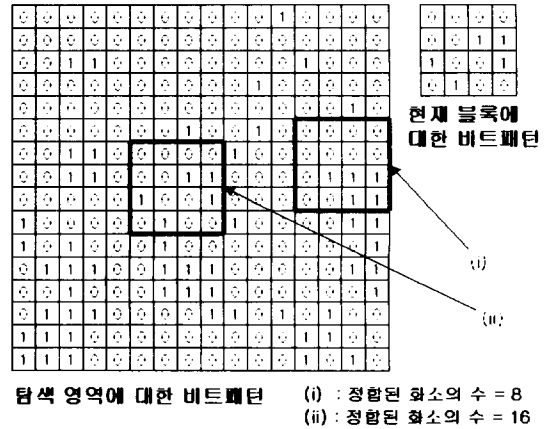
예를 들어, 1bit($n=1$)와 8bit($n=8$))인 경우에 각각 연산의 수는 $N(1)=95$, $N(8)=542$이다. 본 논문에서 비트패턴을 기반으로 제시한 블록 정합 과정은 다음과 같다.

- 단계 1 : 현재 블록과 그 블록에 대한 탐색영역의 비트패턴 생성.
- 단계 2 : 현재 블록의 비트패턴과 탐색영역 내의 후보 블록들의 비트패턴 비교.
- 단계 3 : 가장 많은 수의 bit가 일치된 상위 m개의 후보 블록들의 움직임 벡터 저장.
- 단계 4 : 저장된 후보 블록들에 대한 DBD를 계산하여 최소의 DBD값을 가진 블록을 정합된 블록으로 결정.

여기서 m값의 선택에 따라 계산량과 예측 에러에 영향을 준다. 그림 2에서는 비트패턴을 기반으로 한 블록 정합의 예를 보여주고 있다.

그림 2. 비트패턴을 기반으로 한 블록 정합

3. 블록내의 움직임 정보에 따른 적응적 정합

각각의 블록에 대해 일관된 움직임 예측을 수행하는 것은 정합의 속도에서나 예측의 정확도에서나 효율적인 방식이 아니다. 빠른 시간 내에 정확한 움직임 추정을 하기 위해서는 블록내의 움직임 정보에 따른 적응적 예측 방식이 요구된다. 본 논문에서는 정합 과정의 계산 속도와 예측 에러 감소 측면의 효율성을 모두 고려해 블록 내의 움직임 정보에 따라 적응적 탐색이 이루어지도록 하였다.

3.1 움직임이 없는 블록 결정

일반적으로 동영상의 각각의 프레임은 움직임을 가지는 움직임 객체 부분과 움직임이 거의 없는 배경부분을 가지고 있다. 배경부분에 대한 움직임 예측 과정을 움직임 객체 부분에서와 동일하게 수행한다면 많은 불필요한 탐색과정을 초래한다. 따라서 움직임을 예측하기에 앞서 움직임의 유무를 먼저 결정할 필요가 있다. 이는 식 (4)에서와 같이 움직임 벡터를 (0,0)으로 설정하고 DBD를 계산함으로써 결정할 수 있다.

$$DBD(0,0) = \frac{1}{N \times N} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |F_n(i,j) - F_{n-1}(i,j)|$$
$$\begin{cases} \text{if } DBD(0,0) \le TH_{no-motion}, & \textit{No motion searching} \\ \textit{otherwise}, & \textit{Motion searching} \end{cases} \tag{4}$$

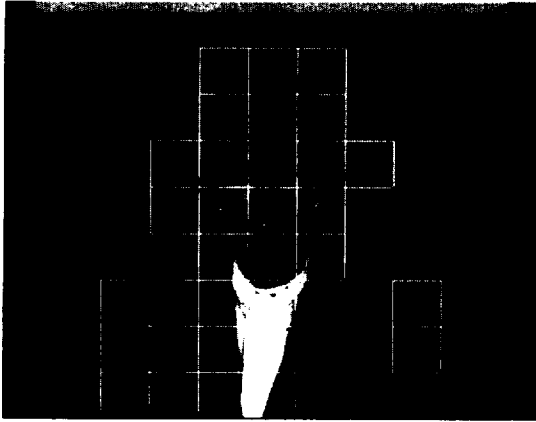
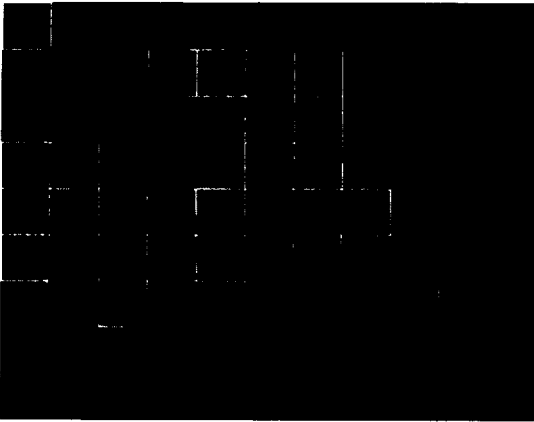
즉, DBD(0,0)이 특정 임계값보다 작으면 움직임이 없다고 추정할 수 있으므로 그 블록에 대해서는 움직임 예측 과정에서 제외시킨다[7]. 이는 탐색 초기에 움직임이 없는 블록에 대한 불필요한 정합과정을 제거시킴으로써 많은 계산적 부담을 줄일 수 있다. 움직임이 있는 블록에 대해서는 움직임 정도에 따라 움직임 예측을 수행하게 된다. 그림 3은 claire와 salesman영상에서 움직임이 있다고 판명된 부분의 블록 형성을 나타낸다. 블록의 수가 전체 영상에서

(a) Claire

(b) Salesman

그림 3. 움직임을 가진 블록 결정

형성할 수 있는 블록 수의 1/3이하인 것을 알 수 있는데, 이는 영상의 2/3이상에서 움직임 추정이 수행되지 않음을 의미한다.

3.2 가변 블록 크기 움직임 추정

가변 블록 크기 움직임 예측은 에러가 많이 발생한 블록에서 보다 정확한 움직임 예측을 하기 위해 사용한다. 정합된 블록과의 DBD값이 특정 임계값보다 클 경우, 식 (5)를 통해 블록을 분할한다.

$$\begin{cases} \text{if} \quad DBD(i,j) \geq TH_{split}, & Split \\ otherwise, & Not\ split \end{cases} \quad (5)$$

분할된 블록의 재탐색 후, DBD값이 다시 임계값보다 클 경우는 재분할을 수행한다. 그림 4는 분할에 따른 각각의 블록레벨(B_{S0}, B_{S1}, B_{S2}, B_{S3})의 블록 분할과 블록들의 크기를 나타낸다. 이러한 분할 방식은 예측 에러가 많은 부분, 즉, 영상 내부의 움직임이 큰 부분에서 정합과정을 심화시키는 것을 의미한다. 이는 예측 에러를 상당량 감소시켜 화질을 개선하는 한편 영상의 국부적 변화에 효과적으로 대응한다. 그림 5는 claire와 salesman영상에 가변 블록 크기를 적용하여 블록 내의 움직임 정보에 따라 블록이 형성되는 모습을 나타내고 있다.

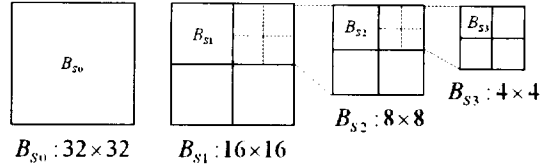



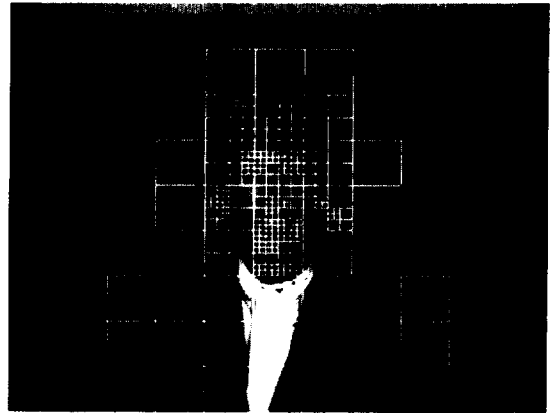
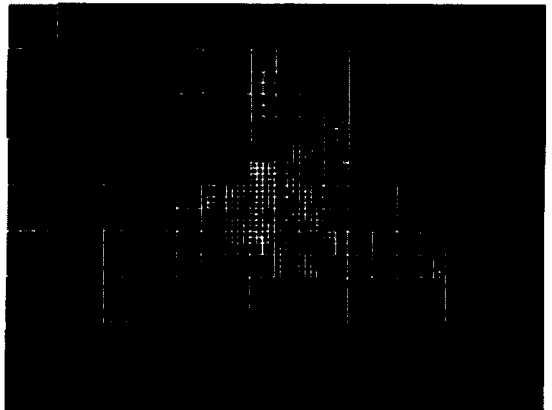
그림 4. 블록 레벨에 따른 블록 형성

(a) Claire

(b) Salesman

그림 5. 움직임 정보에 따른 블록 분할

3.3 이차 탐색 영역 설정

정합된 블록의 움직임 벡터가 블록의 최대 변위인 ($\pm w$, $\pm w$)이라면, 움직임 벡터가 지시하는 곳은 탐색영역의 경계(boundary)이다. 이는 탐색영역의 외부에 정합 블록이 존재할 수도 있음을 의미한다. 정합된 블록이 탐색영역 내부에 존재한다면 그 블록의 DBD값에 따라 정합과정이 종료되거나 블록의 분할과정으로 들어가게 된다. 그러나 움직임 벡터가 탐색영역의 경계를 지시하면 처음 탐색영역 크기의 절반에 해당하는 이차 탐색 영역을 설정하고 재탐색을 수행한다[8]. 이것은 작은 일차 탐색영역의 설정으로 계산적 부담을 줄이고, 움직임이 큰 경우는 이차 탐색영역에서 찾아질 수 있도록 한다. 본 논문에서는 이차 탐색영역의 설정을 블록의 하위레벨인 B_{S1}, B_{S2}, B_{S3} 블록 레벨에서 적용하였다. 그림 6은 일차 탐색 과정에서의 움직임 벡터가 일차 탐색영역의 경계를 지시하여 이차 탐색영역을 설정하고 블록 정합을 수행하는 모습을 나타낸다. 따라서 움직임 벡터(motion vector; MV)는 식 (6)과 같이 일차, 이차 탐색 과정에서 얻어진 움직임 벡터의 합으로 얻어진다.

$$MV = MV_{first} + MV_{second} \quad (6)$$

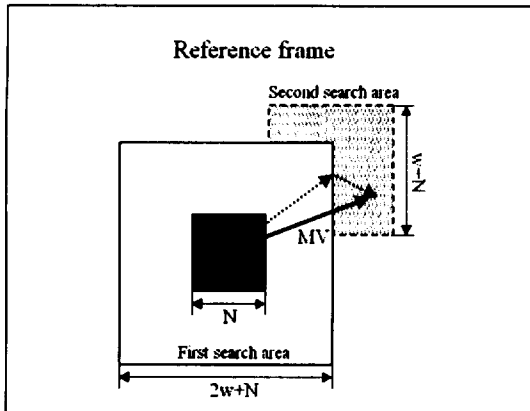



그림 6. 이차 탐색영역

3.4 탐색영역의 위치, 화소 subsampling, 탐색영역의 크기 설정

탐색영역의 위치설정은 정합블록이 존재할 가능성이 높은 곳에 설정해야 한다. 이는 블록 정합의 가능성을 높여주고 탐색영역의 크기를 줄여 탐색과정의 계산적 부담을 줄여줄 수 있다. 움직임 객체내의 움직임 정보는 부드럽고 완만하게 변화한다. 그러므로 움직임 벡터의 불연속은 다른 방향으로 움직이는 객체들의 경계에서만 발생한다. 움직임 객체는 대체로 여러 블록들에 걸쳐 존재하므로 움직임 벡터들은 이웃하는 블록들간에 높은 상관관계가 있다. 따라서 탐색영역의 위치설정은 그림 7에서와 같이 이미 구해진 인접 블록들의 움직임 벡터를 이용한다. 인접 블록들의 움직임 벡터로 탐색 블록의 4개의 DBD를 계산하고 자신의 위치에서의 DBD를 계산한 후 가장 작은 DBD값을 나타낸 움직임 벡터를 탐색영역의 위치로 설정한다. 이것은 가장 상위 레벨 (B_{S0})의 블록인 경우이고, 하위 레벨 (B_{S1}, B_{S2}, B_{S3})의 블록인 경우 상위 레벨에서 얻어진 움직임 벡터를 이용한다. 따라서 상위 레벨로부터의 움직임 벡터의 상속은 작은 탐색영역 설정으로도 큰 움직임 벡터를 얻을 수 있도록 해준다.

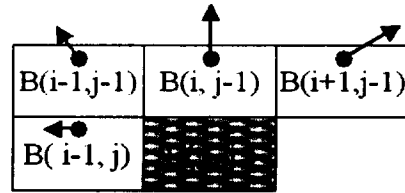



그림 7. 인접한 블록들의 움직임 벡터

FS-BMA에서는 블록간의 유사정도를 측정하기 위해 블록내의 모든 화소를 계산한다. 그러나 블록내 영상정보의 높은 공간적 상관관계의 특성을 이용해 화소 subsampling을 한다면 계산량 절감을 가져올 수 있다. 반면 지나치게 높은 비율의 subsampling은 많은 예측 에러를 발생시킨다. 따라서 블록내의 움직임 정도에 따라 화소의 subsampling 비율을 설정할 필요가 있다. 블록의 크기가 작아질 수록 움직임의 정도는 큰 것을 의미하기 때문에, 그리고 비트 패턴의 단순화를 막기 위해서 하위 레벨의 블록으로 갈수록 낮은 화소 subsampling 비율을 설정한다. 그림 8은 적용될 수 있는 subsampling 패턴의 예를 나

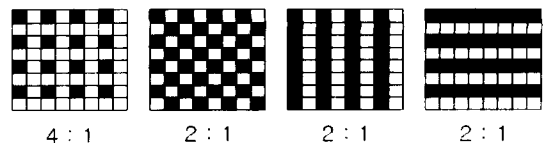



그림 8. 화소 subsampling의 예

타낸다.

본 논문에서 적용한 블록 레벨에 따른 화소 subsampling 비율은 다음과 같다.

- B_{S0} - 4 : 1 pixel-subsampling
- B_{S1} - 2 : 1 pixel-subsampling
- B_{S2}, B_{S3} - No pixel-subsampling

탐색영역의 크기도 화소 subsampling과 같이 블록내의 움직임 정도에 따라 설정한다. 따라서 식 (7)과 같이 움직임이 큰 하위 레벨의 블록으로 갈수록 큰 탐색영역을 설정한다.

$$w_{S0} \leq w_{S1} \leq w_{S2} \leq w_{S3} \qquad (7)$$

여기서 $w_{S0}, w_{S1}, w_{S2}, w_{S3}$는 각각의 블록 레벨에 따라 움직임 벡터가 나타낼 수 있는 최대 변위의 크기를 나타낸다.

3.5 최종 움직임 벡터

각 블록 레벨의 움직임 벡터는 식 (6)에서 얻어진다. 따라서 최종 움직임 벡터는 식 (8)과 같이 각 레벨의 움직임 벡터의 합으로 표현된다.

$$MV_{final} = MV_{S0} + MV_{S1} + MV_{S2} + MV_{S3} \qquad (8)$$

4. 실험 결과 및 고찰

본 논문에서는 알고리즘의 성능평가를 위해 Claire 영상(CIF : 352×288) 1번부터 163번 프레임과 Salesman영상(CIF : 352×288) 1번부터 163번 프레임을 사용하였다. 움직임 예측은 일반적인 동영상 코딩의 참조 프레임(reference frame)의 주기를 적용하여 세 프레임 간격으로 실시하였다. 제안된 알고리즘의 기본 블록 크기는 32×32로 하였고, 4×4까지 분할이 가능하도록 하였다. 탐색영역의 크기 w는 블록의 하위 레벨로 갈수록 탐색영역의 크기(w_{S0}=1, w_{S1}=2, w_{S2}=3, w_{S3}=4)를 증가시켜가며 설정하였다. 따라서 이차 탐색영역까지 고려해 가장 큰 움직임 벡터의 크기는 ±14까지 나올 수 있도록 하였다. 비교 알고리즘으로는 기본 블록 크기가 16×16이고 최대 움직임 벡터의 크기가 ±14(w=14)인 전역 탐색 블록 정합 알고리즘(FS-BMA)을 사용하였다. 본 논문에서는 알고리즘의 비교분석을 위해 실험을 통해 MSE (Mean Square Error)값을 얻어냈고, 식 (9)을 통해 PSNR(Peak Signal to Noise Ration)을 산출하였다.

$$PSNR = 10\log_{10}\frac{255^2}{MSE} \quad [dB] \qquad (9)$$

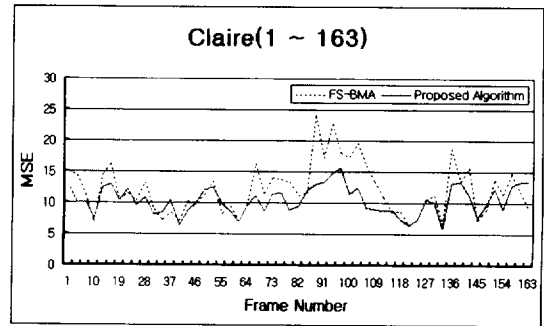
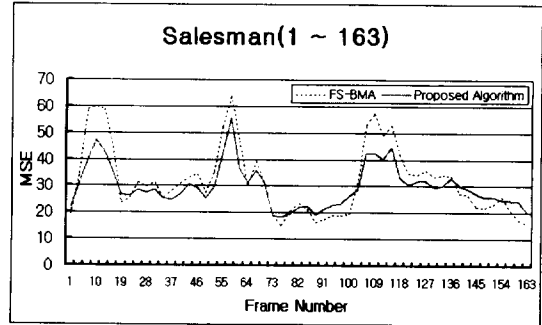
표 1과 그림 9는 제안 알고리즘과 전역 탐색 블록 정합 알고리즘(FS-BMA)과의 비교를 나타낸다. 표 1에서 알 수 있듯이 제안 알고리즘이 FS-BMA에 비해 프레임 당 적은 블록을 가지고 약 0.5dB 정도의 PSNR 개선을 볼 수 있다. 그리고 그림 9에서 제안 알고리즘이 프레임 당 에러값을 적게 발생시키고 프레임간 에러의 편차를 줄여준다는 것을 알 수 있다.

(a) Claire

(b) Salesman

그림 9. Claire와 Salesman 영상에 대한 MSE 비교

표 1. 제안 알고리즘에 대한 성능 평가

	Claire		Salesman	
	FS-BMA	Proposed	FS-BMA	Proposed
상대 연산수	1	0.01	1	0.017
MSE/pixel	12.19	10.44	32.36	29.93
PSNR	37.27	37.94	33.03	33.37
블럭수/Frame	396	178.6	396	297.2

알고리즘의 계산 복잡도는 예측에 소요되는 연산의 수를 산출함으로써 비교하였다[9]. 한 개의 화소를 비교하기 위해서는 3개의 연산(뺄셈, 절대값, 덧셈)이 필요하다. FS-BMA의 경우 블록의 크기가 16×16(=256)이고, w = 14인 탐색 영역 ($(2w+1)^2$=841)일 때, 그 블록에 대한 움직임 벡터를 얻기 위해서는 841×(256×3+1)의 연산이 필요하다. 여기서 1은 DBD 계산 후의 임계값과의 비교 연산이다. 결국, CIF 영상(396블록) 한 프레임을 예측하기 위한 연산의 수는 841×(256×3+1)×396 = 256,104,684 이다. 표 1에서는 전역 탐색의 연산의 수를 1이라 할 때 제안 알고리즘의 상대 연산수를 나타낸다. 제안 알고리즘이 상대적으로 아주 적은 수의 연산을 나타낼 수 있었던 것은 비트패턴을 도입하여 화소의 값을 8bit에서 1bit로 줄인 후, 움직임이 없는 부분은 정합 과정에서 제외하고, 움직임의 크기에 따른 탐색영역 설정과 화소 subsampling을 적용했기 때문이다. 무엇보다 움직임 벡터의 하위레벨로의 상속으로 인해 작은 탐색영역 설정으로도 큰 움직임 벡터값을 얻을 수 있었던 것이 연산량을 줄이는 가장 큰 역할을 했다.

5. 결 론

본 논문에서는 비트패턴을 기반으로 가변 블록 크기 및 여러 적응적 정합 방식을 이용하여 한 프레임당 적은 수의 블록을 가지고 적은 예측 에러를 발생시킬 수 있는 새로운 고속의 움직임 추정 방법을 제안하였다. FS-BMA의 일관된 블록의 크기 및 일관된 탐색영역 설정으로 인한 계산적 부담을 제안 알고리즘에서는 비트패턴을 기반으로 영상내의 움직임 정도에 따라 적응적 정합을 수행하여 불필요한 탐색 과정을 제거하고 고속의 정합이 가능하도록 하였다. 그리고 블록의 크기를 고정으로 했을 때 효율적으로 처리할 수 없는 영상의 국부적 많은 움짐임에 대해서는 블록의 분할을 시도하여 효과적으로 처리하였다. 결국 FS-BMA보다 더 높은 PSNR을 가져왔다. 따라서 제안 알고리즘은 고속의 실시간 처리를 요구하고 영상 내의 움직임의 변화가 적은 화상 회의나 화상전화와 같은 저비트율 전송을 위한 움직임 예측에 효과적으로 적용 할 수 있다.

제안 알고리즘은 정합 과정의 대부분에서 적응적 방식을 적용했지만 임계값의 경우 모든 프레임에서 동일하게 적용하였다. 그러나 이것은 매 프레임마다 움직임의 정도가 다르기 때문에 효율적인 방식이 아니다. 각각의 프레임마다 움직임 정도에 따른 임계값을 얻을 수 있다면 보다 나은 결과를 얻을 것이다. 따라서 움직임 예측을 수행하기 전에 프레임별 최적의 임계값을 얻어낼 수 있는 전처리 과정에 대한 연구가 뒤따라야 할 것이다.

참 고 문 헌

[1] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," IEEE Trans. Communication, vol. COM-29, no. 12, pp. 1799-1808, December 1981.

[2] S. Kappagantula, and K. R. Rao, "Motion compensated interframe image prediction," IEEE Trans. Communication, vol. COM-33, pp. 1011-1015, 1985.

[3] H. Gharavi, and M. Mills, "Block matching motion estimation algorithms - new results," IEEE Trans. Circuits Syst., pp. 649-651, 1990.

[4] J. Zhang, M. O. Ahmad, and M. N. S. swamy, "A New Variable Size Block Motion Compensation," Proceedings of the IEEE Int. Conf. on Image Processing, vol. 2, pp. 164-167, 1997.

[5] J. Feng, K.-T. Mehrpour, and A.E. F. Karbowiak, "Adaptive block matching algorithm for video compression," IEEE Proc. Vision, Image and Signal Processing, vol. 145 no. 3, pp. 173-178, 1998.

[6] H. Kiya, J. Furukawa, and Y. Noguchi, "Block Matching Motion Estimation Based on Median Cut Quantization for MPEG Video," IEICE Trans. Electronics Communications & Computer Sciences, vol. E82-A, no. 6, pp. 899-904, 1999.

[7] 김진한외, "복잡한 움직임 영역의 예측을 통한 가변 블록 크기 움직임 추정", 신호처리 합동학술대회 논문집, 제12권 제 1호, pp. 65-68, 1999. 10. 2.

[8] H. S. Oh, C. H. Lee, H. K. Lee, and J. H. Jeon,

"A new block-matching algorithm based on an adaptive search area adjustment using spatio-temporal correlation," IEEE Trans. Consumer Electronics, vol. 45 no. 3, pp. 745-752, 1999.

[9] J. Chalidabhongse, and J. Kuo, "Fast motion vector estimation using multiresolution spatio-temporal correlations," IEEE Trans. Circuits and Systems for Video Technology, vol. 7 no. 3, pp. 477-488, 1997.

신 동 식

1999년 충북대학교 정보통신공학과 졸업(공학사)
1999년 ~ 현재 충북대학교 정보통신공학과 석사과정
관심분야 : 비디오코딩, 영상처리, 데이터 압축
E-mail : dsshin98@hotmail.com

안 재 형

1981년 충북대학교 전기공학과 졸업(공학사)
1983년 한국과학기술원 전기 및 전자공학과(공학석사)
1992년 한국과학기술원 전기 및 전자공학과(공학박사)
1987년 ~ 현재 충북대학교 전기전자공학부 교수
관심분야 : 영상통신 및 영상정보처리, 멀티미디어 제작 및 정보제공, 인터넷 통신 및 프로그래밍
email : jhahn@cbucc.chungbuk.ac.kr